Udacity Deep Reinforcement Learning: Competition and collaboration project

Implementation

For this project we implemented a multi-agent algorithm that consists of 2 DDPG agents. With this method we increase efficiency by having a faster distribution of the reward signal by having information from the actor and the critic during the training steps, and using only the actor's information during testing. The implementation also overcomes overestimation by sampling the experiences learned with replay buffer.

The submission consists on 1 file, Tennis.ipynb. This file Contains the Actor and Critic classes, both containing a Target and a Local Neural Network for training, the DDPG agent, a Noise (Ornstein-Uhlenbeck process), and a Replay Buffer class.

The architecture of the Networks are the following :

Actor:
First layer: input= 24, output= 200
Second layer: input= 200, output= 150
Third layer : input= 150, output= 2

Critic
First layer: input= 52, output= 200
Second layer: input= 200, output= 150
Third layer : input= 150, output= 1

Training the agent:

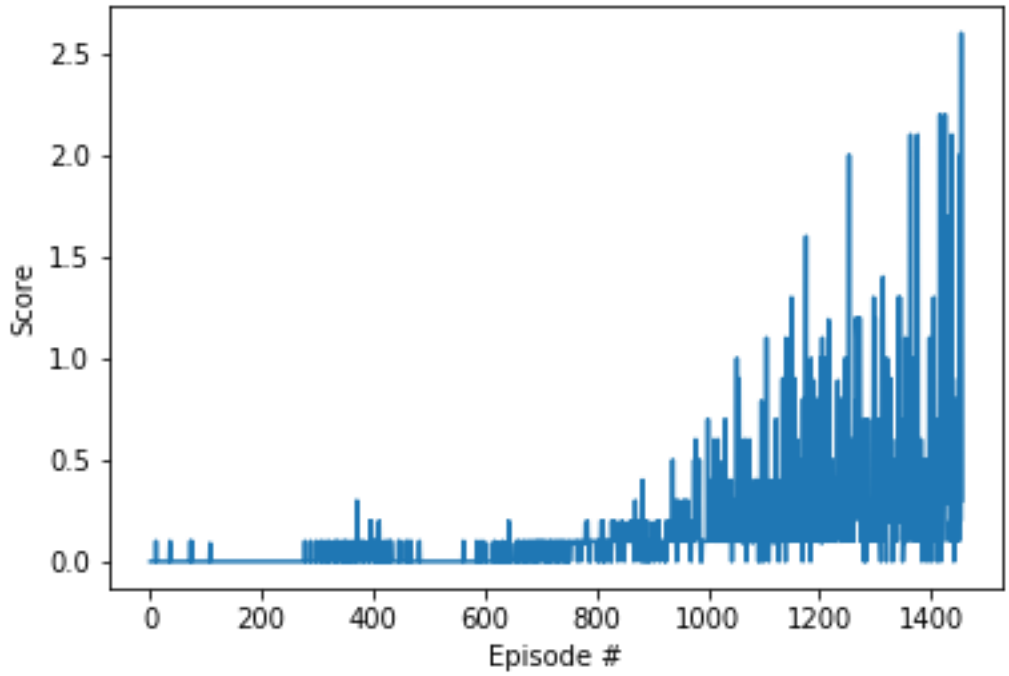
Hyperparameters used to train the agent:

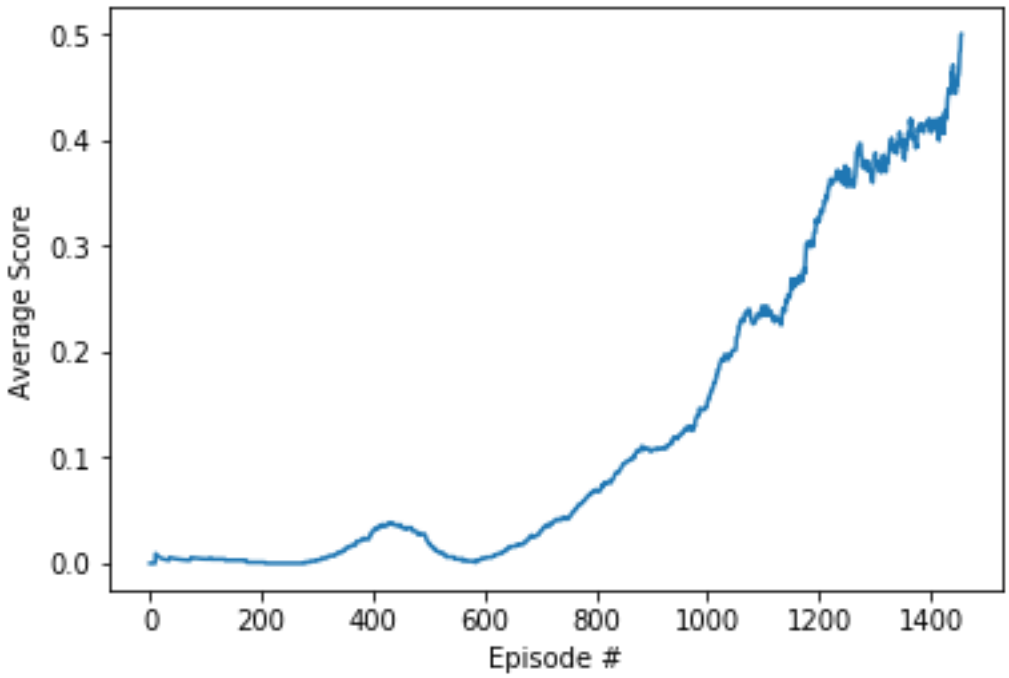
BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 250   # minibatch size
GAMMA = 0.99   # discount factor
TAU = 1e-3   # for soft update of target parameters
LR_ACTOR = 1e-4   # learning rate of the actor
LR_CRITIC = 1e-3   # learning rate of the critic 2539
WEIGHT_DECAY = 0   # L2 weight decay

Results:

```
Episode 100    Average score: 0.004
```

```
Episode 200    Average score: 0.001
Episode 300    Average score: 0.003
Episode 400    Average score: 0.030
Episode 500    Average score: 0.020
Episode 600    Average score: 0.005
Episode 700    Average score: 0.030
Episode 800    Average score: 0.069
Episode 900    Average score: 0.106
Episode 1000   Average score: 0.154
Episode 1100   Average score: 0.236
Episode 1200   Average score: 0.325
Episode 1300   Average score: 0.385
Episode 1400   Average score: 0.408
Solved in episode: 1455      Average score: 0.500
```





Ideas for future work:

Trying out different hyperparameters may help improve the model or maybe reduce the production time. Also multi-agent PPO and multi-agent DQN should be implemented.